

Đã bắt đầu vào lúc	Thứ bảy, 19 Tháng mười một 2022, 9:37 AM
Tình trạng	Đã hoàn thành
Hoàn thành vào lúc	Thứ bảy, 19 Tháng mười một 2022, 10:17 AM
Thời gian thực hiện	39 phút 34 giây
Điểm	3,00/3,00
Điểm	10,00 của 10,00 (100%)



Câu hỏi 1

Chính xác

Điểm 1,00 của 1,00

In this question, you have to perform **add** on AVL tree. Note that:

- When adding a node which has the same value as parent node, add it in the **right sub tree**.

Your task is to implement function: **insert**. You could define one or more functions to achieve this task.



```

#include <iostream>
#include <math.h>
#include <queue>
using namespace std;
#define SEPARATOR "<ab@17943918#@>#"

enum BalanceValue
{
    LH = -1,
    EH = 0,
    RH = 1
};

void printNSpace(int n)
{
    for (int i = 0; i < n - 1; i++)
        cout << " ";
}

void printInteger(int &n)
{
    cout << n << " ";
}

template<class T>
class AVLTree
{
public:
    class Node;
private:
    Node *root;
protected:
    int getHeightRec(Node *node)
    {
        if (node == NULL)
            return 0;
        int lh = this->getHeightRec(node->pLeft);
        int rh = this->getHeightRec(node->pRight);
        return (lh > rh ? lh : rh) + 1;
    }
public:
    AVLTree() : root(nullptr) {}
    ~AVLTree(){}
    int getHeight()
    {
        return this->getHeightRec(this->root);
    }
    void printTreeStructure()
    {
        int height = this->getHeight();
        if (this->root == NULL)
        {
            cout << "NULL\n";
            return;
        }
        queue<Node *> q;
        q.push(root);
        Node *temp;
        int count = 0;
        int maxNode = 1;
        int level = 0;
        int space = pow(2, height);
        printNSpace(space / 2);
        while (!q.empty())
        {
            temp = q.front();
            q.pop();
            if (temp == NULL)
            {

```



```

        cout << " ";
        q.push(NULL);
        q.push(NULL);
    }
    else
    {
        cout << temp->data;
        q.push(temp->pLeft);
        q.push(temp->pRight);
    }
    printNSpace(space);
    count++;
    if (count == maxNode)
    {
        cout << endl;
        count = 0;
        maxNode *= 2;
        level++;
        space /= 2;
        printNSpace(space / 2);
    }
    if (level == height)
        return;
}

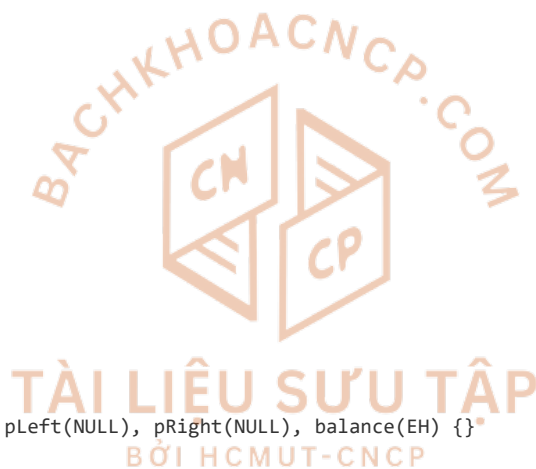
}

void insert(const T &value)
{
    //TODO
}

class Node
{
private:
    T data;
    Node *pLeft, *pRight;
    BalanceValue balance;
    friend class AVLTree<T>;

public:
    Node(T value) : data(value), pLeft(NULL), pRight(NULL), balance(EH) {}
    ~Node() {}
};
};

```



For example:

Test	Result
<pre> AVLTree<int> avl; for (int i = 0; i < 9; i++){ avl.insert(i); } avl.printTreeStructure(); </pre>	<pre> 3 / \ 1 5 / \ / \ 0 2 4 7 \ \ 6 8 </pre>
<pre> AVLTree<int> avl; for (int i = 10; i >= 0; i--){ avl.insert(i); } avl.printTreeStructure(); </pre>	<pre> 7 / \ 3 9 / \ / \ 1 5 8 10 / \ 0 2 4 6 </pre>

Answer: (penalty regime: 0 %)

Reset answer

```

1 //Helping functions
2 Node* rotatepRight(Node *&node)
3 {
4     Node* tmp = node->pLeft;

```

```

5     node->pLeft = tmp->pRight;
6     tmp->pRight = node;
7     return tmp;
8 }
9 Node* rotatepLeft(Node *&node)
10 {
11     Node* tmp = node->pRight;
12     node->pRight = tmp->pLeft;
13     tmp->pLeft = node;
14     return tmp;
15 }
16 }
17
18 Node* pRightBalance(Node *&node, bool &taller)
19 {
20     if (node->pRight->balance == RH) {
21         node->balance = EH;
22         node->pRight->balance = EH;
23         node = rotatepLeft(node);
24         taller = false;
25         return node;
26     }
27     //lor
28     Node* pLeftSubTree = node->pRight->pLeft;
29     if (pLeftSubTree->balance == LH) {
30         node->balance = EH;
31         node->pRight->balance = RH;
32     }
33     else if (pLeftSubTree->balance == EH) {
34         node->balance = EH;
35         node->pRight->balance = EH;
36     }
37     else {
38         node->balance = LH;
39         node->pRight->balance = EH;
40     }
41     pLeftSubTree->balance = EH;
42     node->pRight = rotatepRight(node->pRight);
43     node = rotatepLeft(node);
44     taller = false;
45     return node;
46 }
47
48 Node* pLeftBalance(Node *&node, bool &taller)
49 {
50     //lol
51     if (node->pLeft->balance == LH) {
52         node->balance = EH;
53         node->pLeft->balance = EH;
54         node = rotatepRight(node);
55         taller = false;
56         return node;
57     }
58     //rol
59     Node* pRightSubTree = node->pLeft->pRight;
60     if (pRightSubTree->balance == LH) {
61         node->balance = RH;

```

	Test	Expected	Got	
✓	<pre> AVLTree<int> avl; for (int i = 0; i < 9; i++){ avl.insert(i); } avl.printTreeStructure(); </pre>	<pre> 3 1 5 0 2 4 7 6 8 </pre>	<pre> 3 1 5 0 2 4 7 6 8 </pre>	✓

	Test	Expected	Got	
✓	<pre>AVLTree<int> avl; for (int i = 10; i >= 0; i--){ \tavl.insert(i); } avl.printTreeStructure();</pre>	<pre> 7 3 9 1 5 8 10 0 2 4 6</pre>	<pre> 7 3 9 1 5 8 10 0 2 4 6</pre>	✓

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.



Câu hỏi 2

Chính xác

Điểm 1,00 của 1,00

In this question, you have to perform **delete on AVL tree**. Note that:

- Provided **insert** function already.

Your task is to implement two functions: **remove**. You could define one or more functions to achieve this task.



```

#include <iostream>
#include <math.h>
#include <queue>
using namespace std;
#define SEPARATOR "<ab@17943918#@>#"

enum BalanceValue
{
    LH = -1,
    EH = 0,
    RH = 1
};

void printNSpace(int n)
{
    for (int i = 0; i < n - 1; i++)
        cout << " ";
}

void printInteger(int &n)
{
    cout << n << " ";
}

template<class T>
class AVLTree
{
public:
    class Node;
private:
    Node *root;
protected:
    int getHeightRec(Node *node)
    {
        if (node == NULL)
            return 0;
        int lh = this->getHeightRec(node->pLeft);
        int rh = this->getHeightRec(node->pRight);
        return (lh > rh ? lh : rh) + 1;
    }
public:
    AVLTree() : root(nullptr) {}
    ~AVLTree(){}
    int getHeight()
    {
        return this->getHeightRec(this->root);
    }
    void printTreeStructure()
    {
        int height = this->getHeight();
        if (this->root == NULL)
        {
            cout << "NULL\n";
            return;
        }
        queue<Node *> q;
        q.push(root);
        Node *temp;
        int count = 0;
        int maxNode = 1;
        int level = 0;
        int space = pow(2, height);
        printNSpace(space / 2);
        while (!q.empty())
        {
            temp = q.front();
            q.pop();
            if (temp == NULL)
            {

```




```

        cout << " ";
        q.push(NULL);
        q.push(NULL);
    }
    else
    {
        cout << temp->data;
        q.push(temp->pLeft);
        q.push(temp->pRight);
    }
    printNSpace(space);
    count++;
    if (count == maxNode)
    {
        cout << endl;
        count = 0;
        maxNode *= 2;
        level++;
        space /= 2;
        printNSpace(space / 2);
    }
    if (level == height)
        return;
}

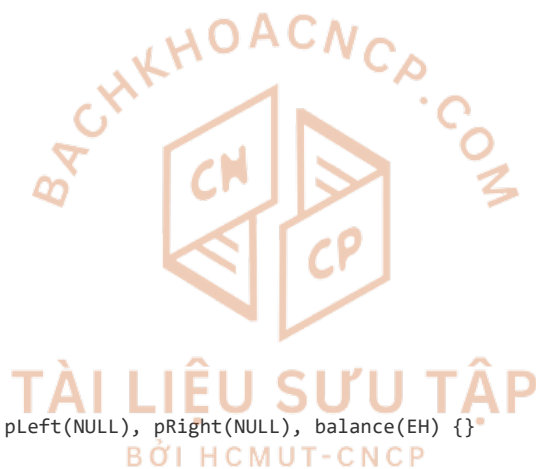
}

void remove(const T &value)
{
    //TODO
}

class Node
{
private:
    T data;
    Node *pLeft, *pRight;
    BalanceValue balance;
    friend class AVLTree<T>;

public:
    Node(T value) : data(value), pLeft(NULL), pRight(NULL), balance(EH) {}
    ~Node() {}
};
};

```

**For example:**

Test	Result
<pre> AVLTree<int> avl; int arr[] = {10,52,98,32,68,92,40,13,42,63}; for (int i = 0; i < 10; i++){ avl.insert(arr[i]); } avl.remove(10); avl.printTreeStructure(); </pre>	<pre> 52 32 92 13 40 68 98 42 63 </pre>
<pre> AVLTree<int> avl; int arr[] = {10,52,98,32,68,92,40,13,42,63,99,100}; for (int i = 0; i < 12; i++){ avl.insert(arr[i]); } avl.remove(13); avl.printTreeStructure(); </pre>	<pre> 52 32 92 10 40 68 99 42 63 98 100 </pre>

Answer: (penalty regime: 0 %)

Reset answer

```

1 //Helping functions
2 Node *rotatepRight(Node *&node)
3 {
4     Node* tmp = node->pLeft;
5     node->pLeft = tmp->pRight;
6     tmp->pRight = node;
7     return tmp;
8 }
9
10
11 Node *rotatepLeft(Node *&node)
12 {
13     Node* tmp = node->pRight;
14     node->pRight = tmp->pLeft;
15     tmp->pLeft = node;
16     return tmp;
17 }
18
19
20
21
22 Node *removepRightBalance(Node *&node, bool &shorter)
23 {
24     if (node->balance == LH) {
25         node->balance = EH;
26         return node;
27     }
28     if (node->balance == EH) {
29         node->balance = RH;
30         shorter = false;
31         return node;
32     }
33     if (node->balance == RH) {
34         //lor
35         if (node->pRight->balance == LH) {
36             Node* pLeftOfpRightTree = node->pRight->pLeft;
37             if (pLeftOfpRightTree->balance == LH) {
38                 node->balance = EH;
39                 node->pRight->balance = RH;
40             }
41             else if (pLeftOfpRightTree->balance == EH) {
42                 node->balance = EH;
43                 node->pRight->balance = EH;
44             }
45             else {
46                 node->balance = LH;
47                 node->pRight->balance = EH;
48             }
49             pLeftOfpRightTree->balance = EH;
50             node->pRight = rotatepRight(node->pRight);
51             node = rotatepLeft(node);
52             //shorter = false;
53         }
54         else if (node->pRight->balance == RH) {
55             node->balance = EH;
56             node->pRight->balance = EH;
57             node = rotatepLeft(node);
58             //shorter = false;
59         }
60         else {
61             node->balance = RH;
62             node->pRight->balance = LH;

```

	Test	Expected	Got	
✓	<pre>AVLTree<int> avl; int arr[] = {10,52,98,32,68,92,40,13,42,63}; for (int i = 0; i < 10; i++){ \tavl.insert(arr[i]); } avl.remove(10); avl.printTreeStructure();</pre>	<pre> 52 32 92 13 40 68 98 42 63</pre>	<pre> 52 32 92 13 40 68 98 42 63</pre>	✓
✓	<pre>AVLTree<int> avl; int arr[] = {10,52,98,32,68,92,40,13,42,63,99,100}; for (int i = 0; i < 12; i++){ \tavl.insert(arr[i]); } avl.remove(13); avl.printTreeStructure();</pre>	<pre> 52 32 92 10 40 68 99 42 63 98 100</pre>	<pre> 52 32 92 10 40 68 99 42 63 98 100</pre>	✓

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.



Câu hỏi 3

Chính xác

Điểm 1,00 của 1,00

In this question, you have to search and print inorder on **AVL tree**. You have to implement functions: **search** and **printInorder** to complete the task. Note that:

- When the tree is null, don't print anything.
- There's a whitespace at the end when print the tree inorder in case the tree is not null.
- When tree contains value, search return true.

```
#include <iostream>
#include <queue>
using namespace std;
#define SEPARATOR "<ab@17943918#@>#"

enum BalanceValue
{
    LH = -1,
    EH = 0,
    RH = 1
};

template<class T>
class AVLTree
{
public:
    class Node;
private:
    Node *root;
public:
    AVLTree() : root(nullptr) {}
    ~AVLTree(){}

    void printInorder(){
        //TODO
    }

    bool search(const T &value){
        //TODO
    }

    class Node
    {
    private:
        T data;
        Node *pLeft, *pRight;
        BalanceValue balance;
        friend class AVLTree<T>;

    public:
        Node(T value) : data(value), pLeft(NULL), pRight(NULL), balance(EH) {}
        ~Node() {}
    };
};
```

For example:

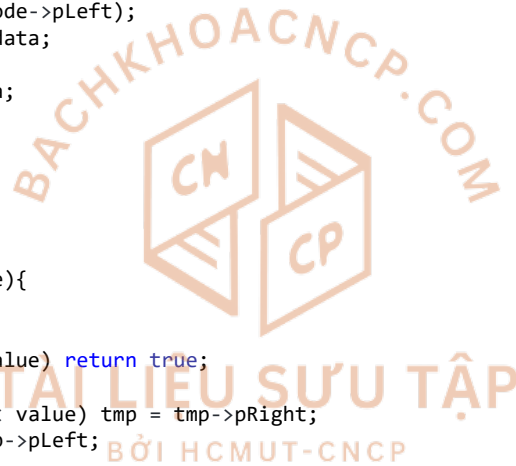


Test	Result
<pre>AVLTree<int> avl; int arr[] = {10,52,98,32,68,92,40,13,42,63,99,100}; for (int i = 0; i < 12; i++){ avl.insert(arr[i]); } avl.printInorder(); cout << endl; cout << avl.search(10);</pre>	<pre>10 13 32 40 42 52 63 68 92 98 99 100 1</pre>

Answer: (penalty regime: 0 %)

```

1 void printInorderRec(Node* node){
2     if(node == nullptr) return;
3     if(node->pLeft && node->pRight){
4         printInorderRec(node->pLeft);
5         cout<<" "<< node->data<<" ";
6         printInorderRec(node->pRight);
7     }
8     else if(node->pRight){
9         cout<< node->data<<" ";
10        printInorderRec(node->pRight);
11    }
12    else if(node->pLeft){
13        printInorderRec(node->pLeft);
14        cout<<" "<<node->data;
15    }
16    else cout<< node->data;
17
18 }
19 void printInorder(){
20     printInorderRec(root);
21 }
22
23 bool search(const T &value){
24     Node* tmp = root;
25     while(tmp){
26         if(tmp->data == value) return true;
27         else{
28             if(tmp->data < value) tmp = tmp->pRight;
29             else tmp = tmp->pLeft;
30         }
31     }
32     return false;
33 }
```



	Test	Expected	Got	
✓	<pre>AVLTree<int> avl; int arr[] = {10,52,98,32,68,92,40,13,42,63,99,100}; for (int i = 0; i < 12; i++){ \tavl.insert(arr[i]); } avl.printInorder(); cout << endl; cout << avl.search(10);</pre>	<pre>10 13 32 40 42 52 63 68 92 98 99 100 1</pre>	<pre>10 13 32 40 42 52 63 68 92 98 99 100 1</pre>	✓

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

BÁCH KHOA E-LEARNING



WEBSITE

HCMUT

MyBK

BKSI

LIÊN HỆ

📍 268 Lý Thường Kiệt, P.14, Q.10, TP.HCM

☎ (028) 38 651 670 - (028) 38 647 256 (Ext: 5258, 5234)

✉ elearning@hcmut.edu.vn



Copyright 2007-2022 BKEL - Phát triển dựa trên Moodle